Mapiranje zahteva

Mapiranje zahteva u Djangu se vrši u fajlu urls.py, koristeći listu urlpatterns, u kojoj su elementi zapravo pozivi funkcije path(). Ta funkcija, kako smo videli iz prethodnih primera, može primati više različitih argumenata, ali je koncept isti – za prispeli korisnički zahtev izvršiti određenu funkcionalnost. Ako tražena putanja nije dostupna, tačnije nije mapirana, klijentu će se pojaviti greška sa kodom 404, što znači da traženi resurs nije pronađen na serveru.

Takođe je važno napomenuti kako Django „zna" u kom fajlu prvenstveno da traži određeni mapirani link. Reč je o promenljivoj ROOT_URLCONF u settings.py fajlu, čija vrednost tipa string predstavlja putanju do prvog urls.py fajla u kojem treba potražiti zatraženi URL. Na primeru našeg projekta first_project vidimo da je vrednost ROOT_URLCONF promenljive zapravo first_project.urls, što znači da će Django prvo tražiti URL u projektnom fajlu urls.py.

U dosadašnjim primerima, u path() funkciji smo definisali samo konkretne zahteve – dakle, ako bi korisnik otvorio stranicu 127.0.0.1/books, dobio bi upravo tu stranicu. Primeri kao što su 127.0.0.1/book ili 127.0.0.1/boo bi prijavili 404 grešku. Nekada se možemo naći u scenariju da imamo listu proizvoda gde svaki proizvod vodi ka stranici sa linkom koji sadrži identifikacioni broj tog proizvoda – na primer, ako bi korisnik otvorio 127.0.0.1/8334/ da mu se prikaže proizvod pod tim identifikacionim brojem. Kada razmišljamo o načinu – kako tu logiku sprovesti i u urls.py fajlu, dolazimo do problema gde bismo za svaki od proizvoda koji postoji u bazi morali da eksplicitno definišemo i mapiramo link. Ovo bi ubrzo postao veliki problem i zato nam Django nudi rešenje – a to je posebna sintaksa koja se koristi pri path() funkciji.

Sintaksa prilikom konvertovanja putanja

Konvertovanje putanja je omogućeno posebnom sintaksom. Ovaj proces se zasniva na ideji da se delovi URL-a segmentiraju po zadatoj logici i da se prema tom rezultatu izvršavaju određene funkcije ili klase pogleda. Zato su konverteri putanja uglavnom vezani za određene tipove podataka kao što su celi brojevi, stringovi i ostali, a lista tih tipova je sledeća:

- str – odgovara bilo kom stringu koji nije prazan ('') bez separatora putanja (/); ovo je podrazumevani konverter putanje;
- int – odgovara bilo kom pozitivnom celom broju uključujući i nulu; vraća Python tip int;
- slug – odgovara poslednjoj sekciji linka na kom se klijent nalazi (www.helloworld.com/hello-world , slug bi bio:'hello-world);
- uuid – odgovara jedinstvenom identifikacionom broju koji se vezuje za modele;
- path – odgovara bilo kom stringu koji nije prazan; omogućava nam pronalaženje tačno zadatih URL-ova, umesto dela, kao što je to slučaj sa int-om ili str-om;

Funkcije za rad sa mapiranjem URL-ova

Funkcije za rad sa mapiranjem URL-ova su:

- path(url putanja, pogled, ime) – vraća element za dalje procesiranje; primer: path('index/', views.index, name='index-view');
- re_path(url putanja, pogled, ime) – vraća element za dalje procesiranje; razlika između ove i path() funkcije je u tome što se re_path() bazira na upotrebi regularnih izraza koje zbog obima i kompleksnosti nisu predmet ovog kursa;
- include(ime modula/fajla) – funkcija kojoj se prosleđuje string tip koji sadrži putanju do sledećeg urls.py fajla koji treba dalje da obradi traženi URL;
- register_converter(konverter, ime konvertera) – funkcija koja se koristi za registrovanje našeg konvertera putanje koji ćemo koristiti u path() funkciji; pošto se za kreiranje naših konvertera putanja koristi posebna sintaksa zajedno sa regularnim izrazima, više o toj temi možete saznati u zvaničnoj dokumentaciji.

Pitanje

Koju promenljivu iz settings.py fajla koristimo za definisanje početne putanje za pretragu zahtevanih linkova?

- WSGI_APPLICATION
- DATABASES
- **ROOT_URLCONF**

Objašnjenje:

Tačan odgovor je da u promenljivoj ROOT_URLCONF podešavamo početnu putanju ka prvom urls.py fajlu u kom treba potražiti zahtevani link.

Primer korišćenja konvertera putanja

Za primer ćemo koristiti projekat iz prethodne nastavne jedinice, s tim što projektni i aplikativni (book_library aplikacija) urls.py sad izgledaju ovako:

first_project/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.views.generic import TemplateView
urlpatterns = [
    # path('', admin.site.urls),
    # path('', views.index, name = 'index')
    # path('',TemplateView.as_view(template_name='index.html')),
    path('', include('book_library.urls')),
]
```

book_library/urls.py

```
from django.contrib import admin
from django.urls import path, include
from . import views
# from django.views.generic import TemplateView
```

```
urlpatterns = [
    # path('', admin.site.urls),
    path('',views.main_page, name = 'main_page'),
    path('books/', views.index, name = 'index_page'),
    path('books/<int:int_key>/', views.int_test, name = 'int_test'),
    path('books/<str:book_name>/',views.index, name = 'str_test'),
]
```

Zakomentarisane linije su linije koje smo koristili u prethodnim verzijama fajla.

Iz ovih primera prvo uviđamo da smo svu logiku za mapiranje URL-ova prebacili iz projektnog u aplikacijski urls.py funkcijom include('book_library.urls'). U fajl book_library/urls.py smo sada već iskoristili konvertere putanja, i to one koji se odnose na stringove i cele brojeve:

- path('books/<int:int_key>/', views.int_test, name = 'int_test'), – ovom linijom smo mapirali sve linkove koji počinju sa books/ a završavaju se pozitivnim celim brojem – books/1, books/1236 itd. Sintaksa konvertera putanje je ista i važi i za ostale, i to po sledećem principu:
    - < > – zagrade koje u stringu označavaju deo u kojem definišemo konverter putanje;
    - <int: > – ključna reč int koja označava tip konvertera putanje;
    - <int:int_key> – nakon imena konvertera putanje koristimo dve tačke, kojima odvajamo konverter sa definisanim, proizvoljnim imenom koje smo dodelili upravo tom tipu za trenutni URL.

Ovo znači da smo svaki broj koji prosledimo nakon books/ dela URL-a nazvali int_key. I po tom imenu int_key mu možemo pristupiti kroz funkcije pogleda. Na ovaj način možemo prosleđivati godine našoj aplikaciji, a klijentu vratiti kao odgovor – knjige izdate u toj godini.

- path('books/<str:book_name>',views.index, name = 'str_test'), – ovom linijom smo mapirali sve linkove koji počinu sa books/ a završavaju se stringom: books/Hamlet, 'books/The Raw Youth' itd. Na isti način, kao i kod int konvertera putanje, iskoristili smo i str tip i nazvali ga book_name. Po tom imenu možemo pristupiti njegovoj vrednosti u funkciji pogleda koja se na ovaj mapirani URL odnosi. Na ovaj način postižemo da od korisnika dobijemo ime knjige a vratimo mu godinu u kojoj je knjiga izdata.

Fajl book_library/views.py je dosta izmenjen u odnosnu na verziju iz prošle nastavne jedinice kako bi ispratio promene u urls.py fajlu i izgleda ovako:

book_library/views.py:

```
from django.shortcuts import render
from django.http import HttpResponse
books = [{'title':'The Picture of Dorian Gray','year':'1890'},
        {'title':'Pride and Prejudice','year':'1813'},
        {'title':'The Adventures of Tom Sawyer', 'year':'1875'},
        {'title':'The Raw Youth', 'year':'1875'},
        {'title':'Twelve Years a Slave', 'year':'1853'},
        {'title':'Hamlet', 'year':'1603'}]
# Create your views here.
```

```
def index(request, book_name = None):

    # return HttpResponse("<h1>Book library</h1>")
    # return HttpResponse(render(request, 'index.html',
{'books':books}))
    if not book_name:
        return render(request, 'index.html', {'books':books})
    temp_res = []
    if book_name:
           for x in books:
            if str(book_name).lower() == x['title'].lower():
temp_res.append(x['year'])

        if temp_res:
            return HttpResponse("<h1>Book {} was published in {}
year.</h1>".format(book_name, temp_res[0]))
        else:
            return HttpResponse("<h1>We couldnt find year published
for book: {}.</h1>".format(book_name))

def main_page(request):
    return HttpResponse("<h1>Welcome to our book library!</h1>")

def int_test(request,int_key):
    temp_res = []
      for x in books:
                  if str(int_key) == x['year']:
                        temp_res.append(x['title'])
    if temp_res:
        return HttpResponse("<h1>You picked year - {}.\
            <br>Book published in that year is:
{}.</h1>".format(int_key, temp_res[0]))
    else:
        return HttpResponse("<h1>There is no book for a given year: {}
in our database.</h1>".format(int_key))
```

Ovde već uviđamo nekoliko ključnih razlika u odnosu na verzije iz prethodnih nastavnih jedinica:

1. Funkciji index(), pored standardnog objekta koji predstavlja korisnikov zahtev, prosleđujemo i book_name argument koji odgovara konverteru putanje. Podrazumevana vrednost je None. Ovo je učinjeno iz razloga jer funkciju index() koristimo u dva slučaja – kada korisnik pristupi books/ putanji (u ovom slučaju je book_name None) i kada korisnik pristupi putanji books/Hamlet (u ovom slučaju book_name uzima vrednost Hamlet). Zato se u funkciji i nalazi logika koja proverava da li vrednost te promenljive, odnosno ime knjige postoji u našoj listi rečnika i na osnovu nje šalje odgovor klijentu (ili knjiga postoji i šalje se godina izdanja ili se šalje odgovor sa porukom da knjiga ne postoji).
2. Funkcija main_page() je dodata kako bi se korisniku na ekranu ispisala poruka – *Welcome to our book library!* nakon što otvori početnu stranu sajta (127.0.0.1:8000).
3. Funkcija int_test(request, int_key) nam služi za pronalaženje knjige koja je izdata u traženoj godini i vraćanje te informacije korisniku. Njoj se prosleđuje standardni objekat zahteva, kao i int_key – imena konvertera putanje koje smo definisali u book_library.views.py fajlu (linija: path('books/<int:int_key>/', views.int_test, name

= 'int_test')). Ova funkcija će se izvršiti ako korisnik nakon books/ doda i ceo broj, pa tako neki od traženih linkova mogu izgledati ovako:

- books/1853 – vratiće poruku *You picked year 1853. Book published in that year is: Twelve Years a Slave*.
- books/2000 – vratiće poruku *There is no book for a given year: 2000 in our database*.

Logika iza book_library/urls.py fajla je takva da ako korisnik otvori početnu stranicu – izvršiće se pogled funkcija main_page(), koja ispisuje samo *Welcome to our book library!*. U slučaju da korisnik otvori books/ stranicu – na ekranu će mu se ispisati lista knjiga koju generiše index() funkcija. Ako bismo prosledili books/Hamlet – dakle, vrednost konvertera putanje book_name postaje Hamlet, na ekranu bismo dobili poruku *Book Hamlet was published in 1603 year*, a u slučaju da je klijent ukucao adresu books/1853 – dobio bi odgovor sa imenom knjige koja je tada objavljena.

Sačuvati sve fajlove i pokrenuti server komandom python manage.py runserrver '127.0.0.1:8000' i posmatrati odgovore servera na sledeće linkove:

- 127.0.0.1:8000
- 127.0.0.1:8000/books
- 127.0.0.1:8000Books
- 127.0.0.1:8000/books/1853
- 127.0.0.1:8000/books/0
- 127.0.0.1:8000/books/Hamlet i 127.0.0.1:8000/books/hamlet
- 127.0.0.1:8000/books/Pride and Prejudice
- 127.0.0.1:8000/books/The Adventures of Tom Sawyer

Rezime

- Mapiranje zahteva u Djangu se vrši u fajlu views.py koristeći listu urlpatterns u kojoj su elementi zapravo pozivi funkcije path().
- Promenljiva ROOT_URLCONF u settings.py fajlu, čija je vrednost tipa string, predstavlja putanju do prvog urls.py fajla u kojem treba potražiti zatraženi URL.
- Konvertovanje putanja je omogućeno posebnom sintaksom. Ovaj proces se zasniva na ideji da se delovi URL-a segmentiraju po zadatoj logici i da se prema tom rezultatu izvršavaju određene funkcije ili klase pogleda. Zato su konverteri putanja uglavnom vezani za određene tipove podataka kao što su celi brojevi, stringovi i ostali.
- Lista tipova dostupnih u konvertovanju putanja je sledeća:
  - str;
  - int;
  - slug;
  - uuid;
  - path.

- Funkcije za rad sa mapiranjem url-ova su:
  - path(url putanja, pogled, ime);

- re_path(url putanja, pogled, ime);
- include(ime modula/fajla);
- register_converter(konverter, ime konvertera).